IN THE SPECIFICATION

Please amend the second paragraph of page 38 as follows:

In addition to being evaluated as to its compliance with the technical requirements, the structure being designed is compared to preexisting technology. This comparison may be by template matching, processing logic, or other means (as detailed below). An aim of using a comparison to pre-existing technology is to design a structure that satisfies the technical requirements of the design problem while simultaneously avoiding preexisting technology (prior art). Thus, processing logic compares the structure to existing technology (processing block ~~143~~147).

Please amend the second paragraph of page 89 as follows:

(3) compare the fully developed circuit to preexisting technology to obtain an isomorphism value of the fully developed circuit using the ladder filter template (processing block ~~143~~ 147 of Figure 1A), and

Please amend from the first paragraph of page 105 to the third paragraph of page 110 as follows:

(1) The Host ~~36~~3050 consists of host processor ~~36~~3001. Host processor ~~36~~3001 has a keyboard ~~36~~3002, a video display monitor ~~36~~3003, and a large disk memory ~~36~~3004. In one embodiment, there are three processes resident on Host processor ~~36~~3001, namely a process for starting a run on each of multiple processors of the system, a process for collecting the

results produced by the processors during the run, and a process for visually displaying and viewing the results being produced during a run.

(2) The group of processing nodes ~~36~~3006 contains multiple processors of the parallel system. Only nine such processors are shown in Figure 30 for reasons of space; however, the examples above were run on parallel systems involving 66 and 1,000 such processing nodes. In one embodiment, there are four processes that are resident on each processing node of the parallel system, namely a Monitor process, a Breeder process, an Exporter process, and an Importer Process. These processors are often referred to as processing nodes of the parallel system.

(3) Communication lines (shown as dotted lines in the figure) between the host ~~36~~3001 and each of the processing nodes of the parallel system. One such communication line ~~36~~3040 is indicated in the figure. In one embodiment, Ethernet is used for these communications lines.

(4) Communication lines (shown as solid lines in the figure) between the various processing nodes of the parallel system. These communications lines are arranged in a 3 by 3 toroidal arrangement in the figure in which each processing node has four neighbors (often called north, east, west, and south). One such communication line ~~36~~3060 is indicated in Figure 30. In one embodiment, Ethernet is used for these communications lines.

(5) In one embodiment, there is a Randomizer ~~36~~3070 is connected by communication lines (shown as dotted lines in Figure 30) to each of processing nodes of the parallel system. In one embodiment, Ethernet is used for these communications lines.

Figure 31 illustrates the four processes resident on each node of one embodiment of a parallel genetic programming system, namely the Monitor process ~~37~~3102, the Breeder process ~~37~~3101, the Exporter process ~~37~~3104, and the Importer Process ~~37~~3103. The primary

process on each of the processing nodes is the Breeder process ~~37~~3101, which executes the bulk of the steps of the search algorithm (genetic programming). The other three processes permit asynchronous communication between the processing nodes of the system. They allow the Breeder process ~~37~~3101 to efficiently exploit the computational power of the processor at each processing node. The Monitor process ~~37~~3102 of a given processing node communicates with the Starter process and Recorder Process of the host computer and the Breeder process of the processing node. The Breeder process ~~37~~3101 also communicates with the Importer process ~~37~~3103 and the Exporter process ~~37~~3104 of the processing node. In one embodiment (in which each processing node has four neighbors), the Exporter process ~~37~~3104 has four buffers for briefly storing emigrants that are about to leave for the four toroidally adjacent processing nodes. Also, the Importer process has four buffers for storing arriving immigrants they (asynchronously) arrive from the four toroidally adjacent processing nodes.

Figure 32 illustrates the processes resident on the host processor of one embodiment of a parallel genetic programming system. In one embodiment, the host processor houses various processes, including the Starter process, the Recorder process, and the Viewer process.

Three Processes Resident on the Host Processor

As shown in Figure 32, the human user ~~24~~3210 initiates a run by sending a command to the Starter process ~~24~~3220 resident on the host processor. This command may be entered on a keyboard ~~36~~3002 (Figure 30) and the control parameters for the run may be supplied from a file (such as ~~36~~3009 in Figure 30).

The Starter process ~~24~~3220 (Figure 32) at the host processor then initiates the run by sending a start-up message to each of the processing nodes ~~24~~3280 (Figure 32) of the parallel system. The start-up message is communicated on the Ethernet and is received by the Monitor process ~~37~~3102 (Figure 31) at each processing node of the parallel system.

The Starter process ~~24~~3220 (Figure 32) also sends a message that starts up a Recorder process ~~24~~3230 on the host processor. After doing this, the Starter process ~~24~~3220 performs no additional function during the run.

As each generation is completed on each processing node of the parallel system, each processing node sends an end-of-generation message to the Recorder process at the host processor. The end-of-generation message is communicated on the Ethernet and is sent by the Monitor process ~~37~~3102 (Figure 31) of the reporting processing node. In an alternate embodiment, these reporting messages are sent less frequently than once per generation per processing node. The Recorder process stores some of the information in the received end-of-generation messages on an output file ~~36~~3004 (Figure 30). In one embodiment, the entire end-of-generation message is stored onto the output file only if the report contains a new best value of fitness (a so-called pace-setting individual) or a new best number of hits.

In one embodiment, there is a Viewer process ~~24~~3230 (Figure 32) associated with the Recorder process of the host processor. The Recorder process starts up the Viewer process. The Recorder process extracts important information from each end-of-generation message and communicates this information to the Viewer process. This information may consist of an indication that the Recorder process has received a message from each processor of the parallel system within a specified time period. For example, if there are 1,000 processors, the Viewer process may present to the human user (by means of a video monitor ~~36~~3003, for example) a 25 by 40 array of colored squares wherein the color of each square indicates that

the Recorder process has received a message from the particular processor within the previous 15 minutes. For example, in one embodiment, a square in this array is green if the Recorder process has received a message from a particular processor within the past 15 minutes; yellow if the Recorder process has received a message from a particular processor between 15 and 60 minutes ago; and red if the Recorder process has not received a message from a particular processor in the past hour.

This 25 by 40 array of colored squares provides the human user with an overall picture of activity in the parallel system and may indicate that a particular processor is not operating.

In a similar manner, the Recorder process may additionally communicate to the Viewer process the best value of fitness achieved on each processing node of the parallel system. A second 25 by 40 array of squares may be presented to the human user, wherein each square contains the best numerical value of fitness achieved at that particular processor of the parallel system. Likewise, another 25 by 40 array of squares may be presented to the human user, wherein each square contains the best number of hits achieved at that particular processor of the parallel system.

The Recorder process may also communicate to the Viewer process other important summary information about the run, including, for example, the best overall value of fitness and the best overall number of hits achieved so far during the run. In one embodiment, this information may be presented as a graph.

The division of functions between the Starter process and the Recorder process enables results to be collected from a run even if the host computer fails momentarily during the run. If the host computer fails during the run, the end-of-generation messages directed to the Recorder process will be lost. However, as soon as the host computer and the Recorder process is restored, the Recorder process begins anew collecting information about the latest

generation of each processing node of the parallel system. In one embodiment, the Recorder process adds information to output file ~~36~~30004 (Figure 30) by incrementing the contents of the file. Thus, although some intermediate information may be lost, the final result of the run will not be lost.

The function of the Starter process and the Recorder process may, in one embodiment, be combined into one process at the host computer. However, it is preferable to divide the above-described functions between the Starter process and the Recorder process since this division of functions means that the continuous operation is not required of the host computer. This division adds a degree of fault-tolerance to the overall parallel system. Additional fault tolerance may be achieved by employing two processors, each containing a Recorder process (with all messages from the processing nodes being sent to both Recorders).

The Starter process sends the initial start-up messages to each of the processing nodes. In one embodiment, at the beginning of a run of genetic programming, the Starter process reads in a set of control parameters from file ~~36~~30009 on Host computer ~~36~~30001. Then, the Starter process sends a start-up message to the Monitor process of each processing node (which will, in turn, be sent by the Monitor process to the Breeder process of that particular processing node).